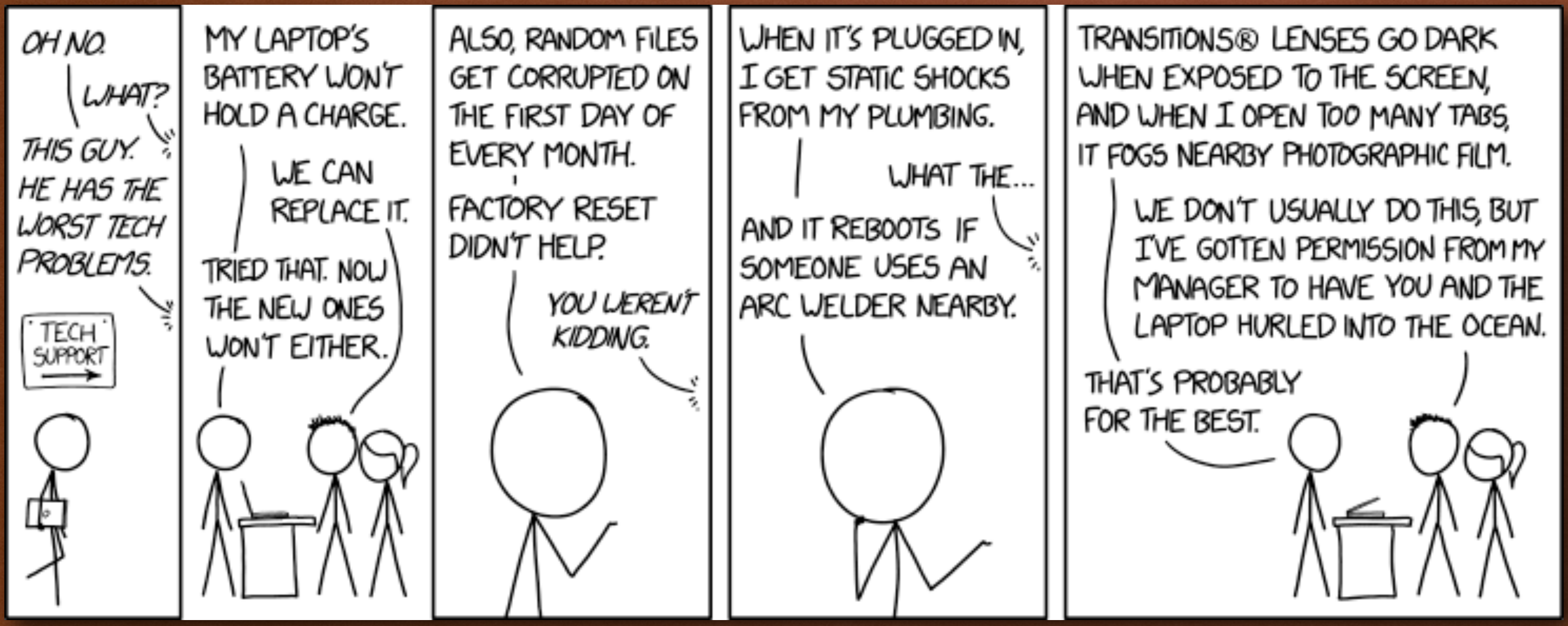
THE BUILDING BLOCKS: COMPUTING WITH GATES



https://xkcd.com/2083/

WHAT HAVE YOU LEARNT SO FAR?

- hexadecimal arithmetic
- binary arithmetic
- logic gate - and, or, not
- encoding of data - signed, unsigned integers, floating point, text, compression, audio, pictures
- boolean expressions
- circuits
- transistors
- truth tables

LEARNING OBJECTIVES

- Designing useful circuits
  - Comparing n-bit values
  - Adding n-bit values
  - Subtracting n-bit values

A USEFUL CIRCUIT

Compare-for-equality (CE) circuit

- Input is two unsigned binary numbers - n-bits
- Output is 1 if inputs are identical and 0 otherwise.
- Start with 1-bit version (1-CE) and build general version from that.

- What is useful about a compare-for-equality circuit?

THE TRUTH TABLE AND EXPRESSION

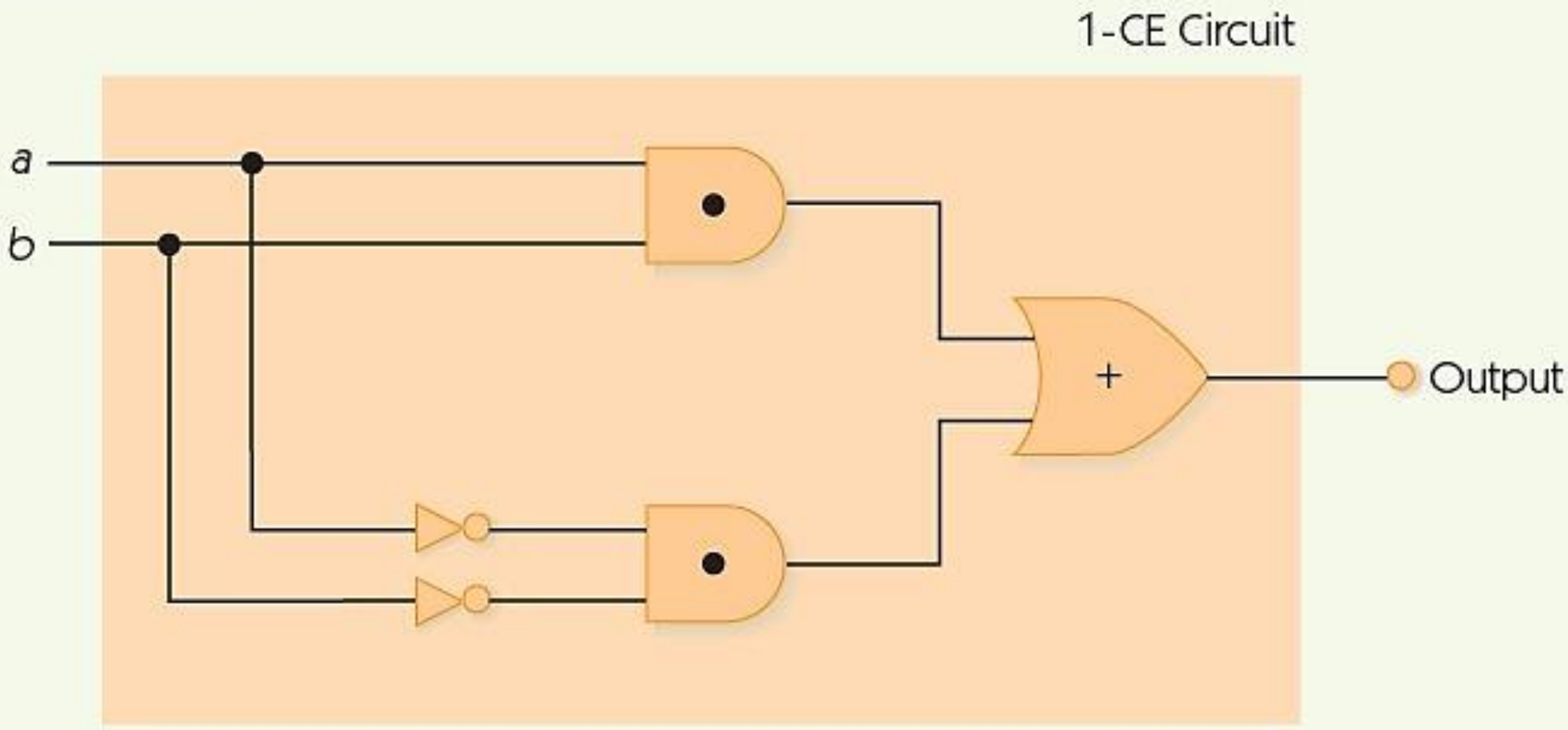
- 1-CE circuit: compare two input bits for equality
- Truth table

a	***b***	***Output***
0	0	1 ← case 1 (both numbers equal to 0)
0	1	0
1	0	0
1	1	1 ← case 2 (both numbers equal to 1)

Boolean expression: $(a \cdot b) + (\overline{a} \cdot \overline{b})$

COMPARE ONE BIT VALUES CIRCUIT

FIGURE 4.27



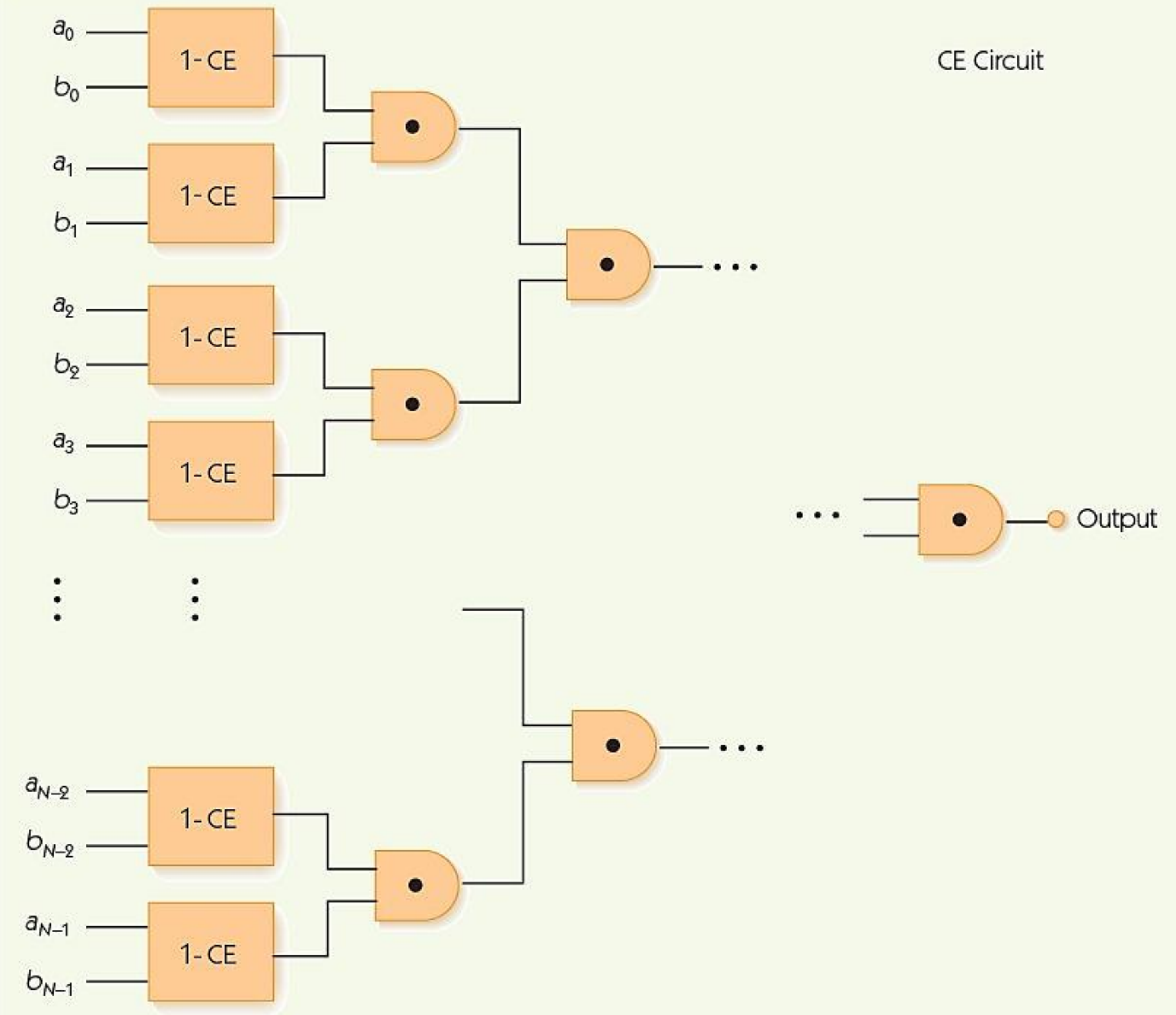
One-bit compare-for-equality circuit

COMBINING LOTS OF BITS

- N-bit CE circuit
- Input: $a_{n-1}...a_2a_1a_0$ and $b_{n-1}...b_2b_1b_0$, where a_i and b_i are individual bits
- Pair up corresponding bits: a_0 with b_0, a_1 with b_1, etc.
- Run a 1-CE circuit on each pair
- AND the results

AN N-BIT COMPARISON CIRCUIT

FIGURE 4.28



N-bit compare-for-equality circuit

ADDING

Full adder circuit

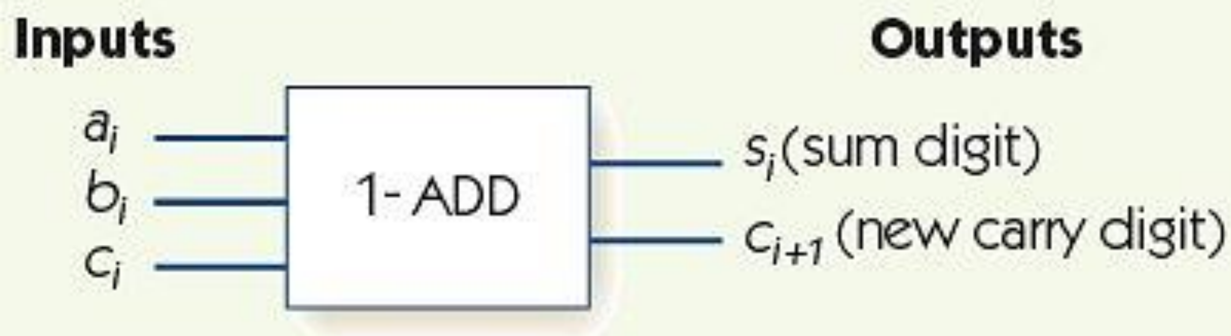
- Input is two unsigned N-bit numbers
- Output is one unsigned N-bit number, the result of adding inputs together
- Example

carry	0	0	0	1	
	0	0	1	0	1
+	0	1	0	0	1
sum	0	1	1	1	0

Start with 1-bit adder (1-ADD)

TRUTH TABLE …

FIGURE 4.29

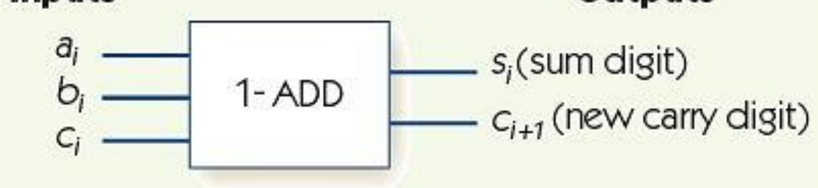


Inputs			Outputs	
a_i	b_i	c_i	s_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

The 1-ADD circuit and truth table

… AND EXPRESSIONS

FIGURE 4.29



Inputs			Outputs	
a_i	b_i	c_i	s_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

The 1-ADD circuit and truth table

Sum digit, s_i, has the Boolean expression

$$(\sim a_i \bullet \sim b_i \bullet c_i) + (\sim a_i \bullet b_i \bullet \sim c_i) + (a_i \bullet \sim b_i \bullet \sim c_i) + (a_i \bullet b_i \bullet c_i)$$

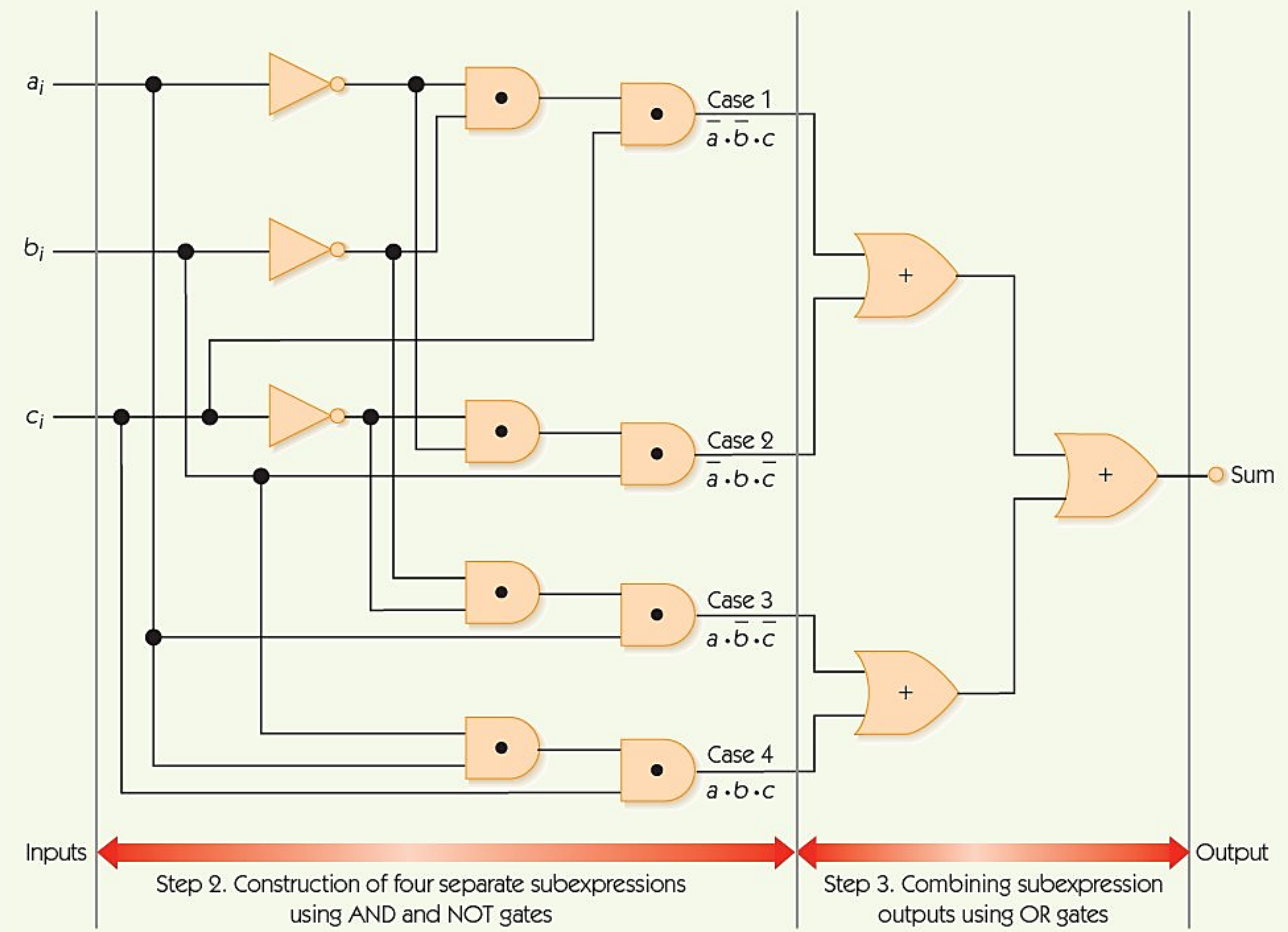
Carry digit, c_{i+1}, has the Boolean expression

$$(\sim a_i \bullet b_i \bullet c_i) + (a_i \bullet \sim b_i \bullet c_i) + (a_i \bullet b_i \bullet \sim c_i) + (a_i \bullet b_i \bullet c_i)$$

The ~ symbol is another way of showing NOT.

TEXTBOOK

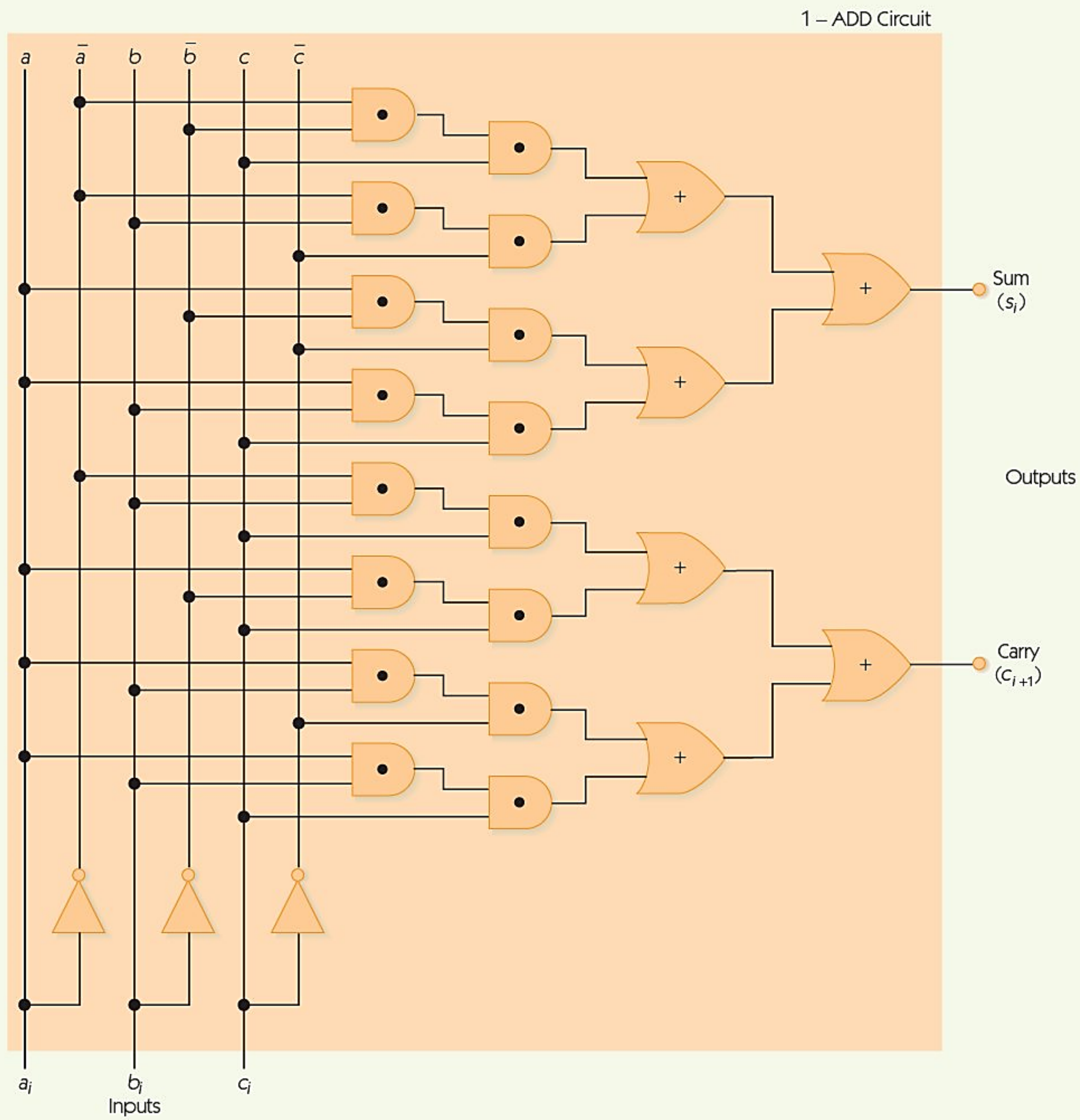
FIGURE 4.30



Sum output for the 1-ADD circuit

TEXTBOOK SUM AND CARRY CIRCUITS

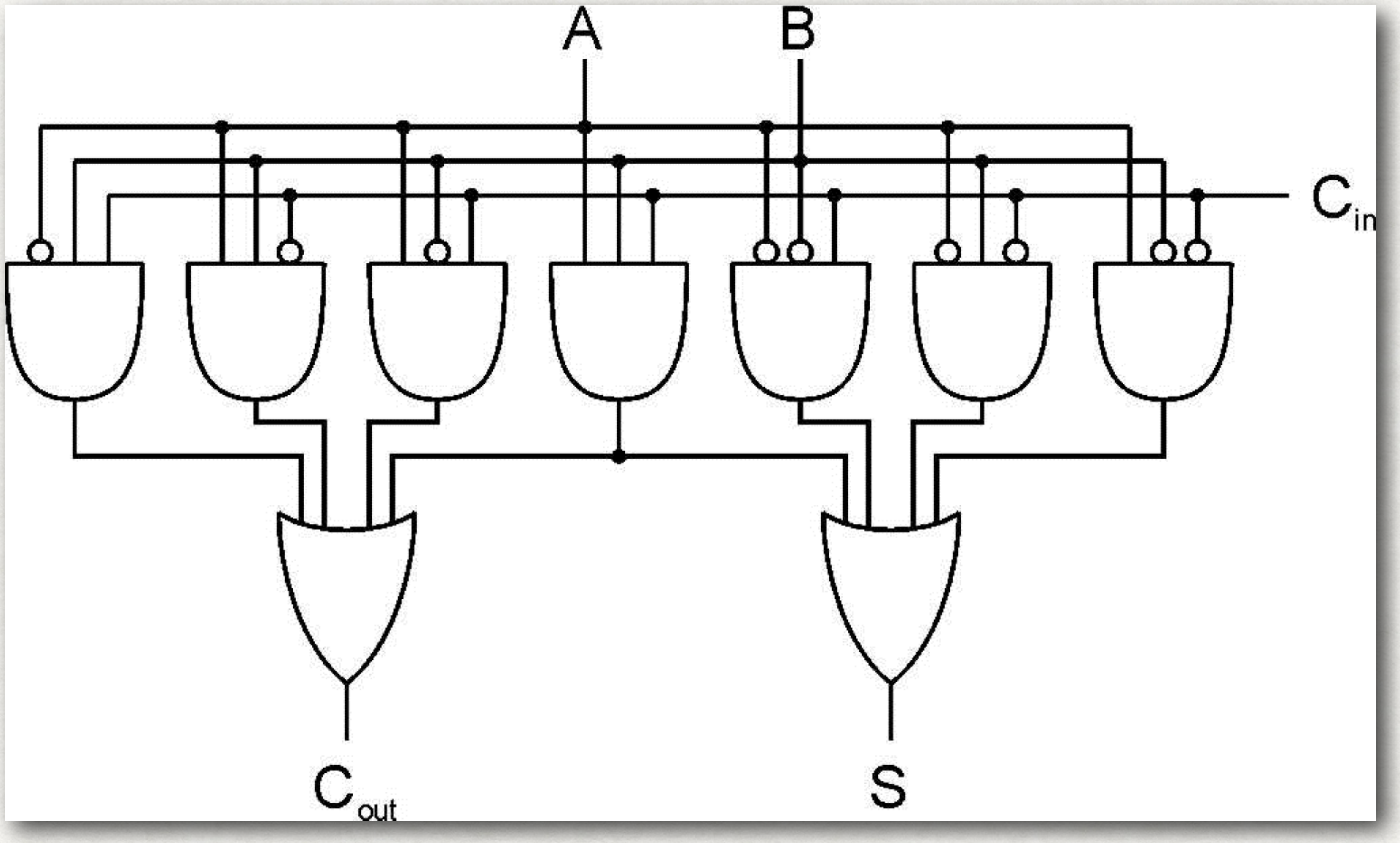
FIGURE 4.31



Complete 1-ADD circuit for 1-bit binary addition

ALTERNATE VERSION

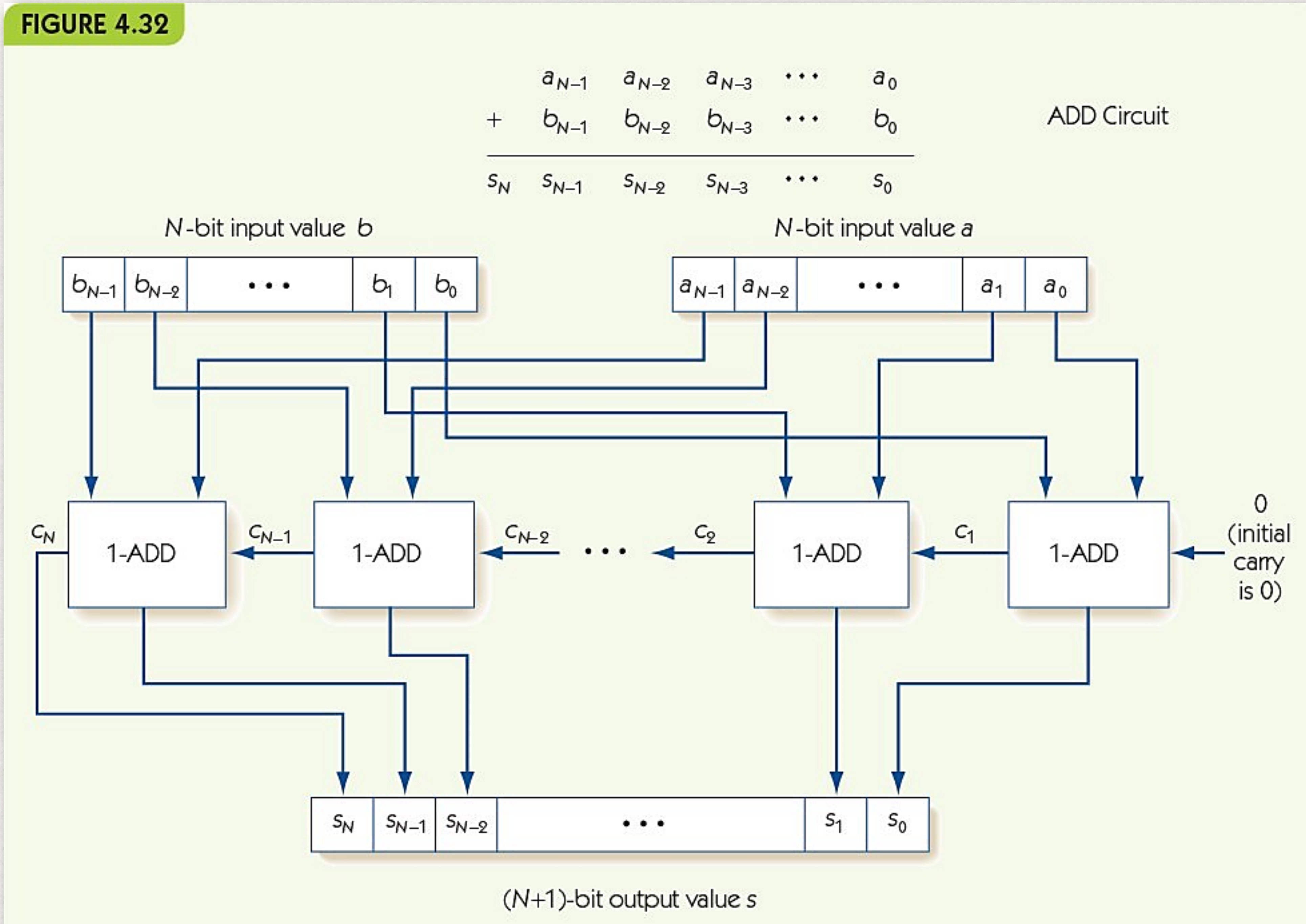
- N.B. The middle AND gate is shared between the carry and sum circuits. (What do those little circles mean?)



FULL ADDER TO N-BITS

- N-bit adder circuit
- Input: $a_{n-1}...a_2a_1a_0$ and $b_{n-1}...b_2b_1b_0$, where a_i and b_i are individual bits
- a_0 and b_0 are least significant digits: ones place
- Pair up corresponding bits: a_0 with b_0, a_1 with b_1, etc.
- Run 1-ADD on a_0 and b_0, with fixed carry in $c_0 = 0$
- Feed carry out c_1 to next 1-ADD and repeat

N-BIT ADDER CIRCUIT

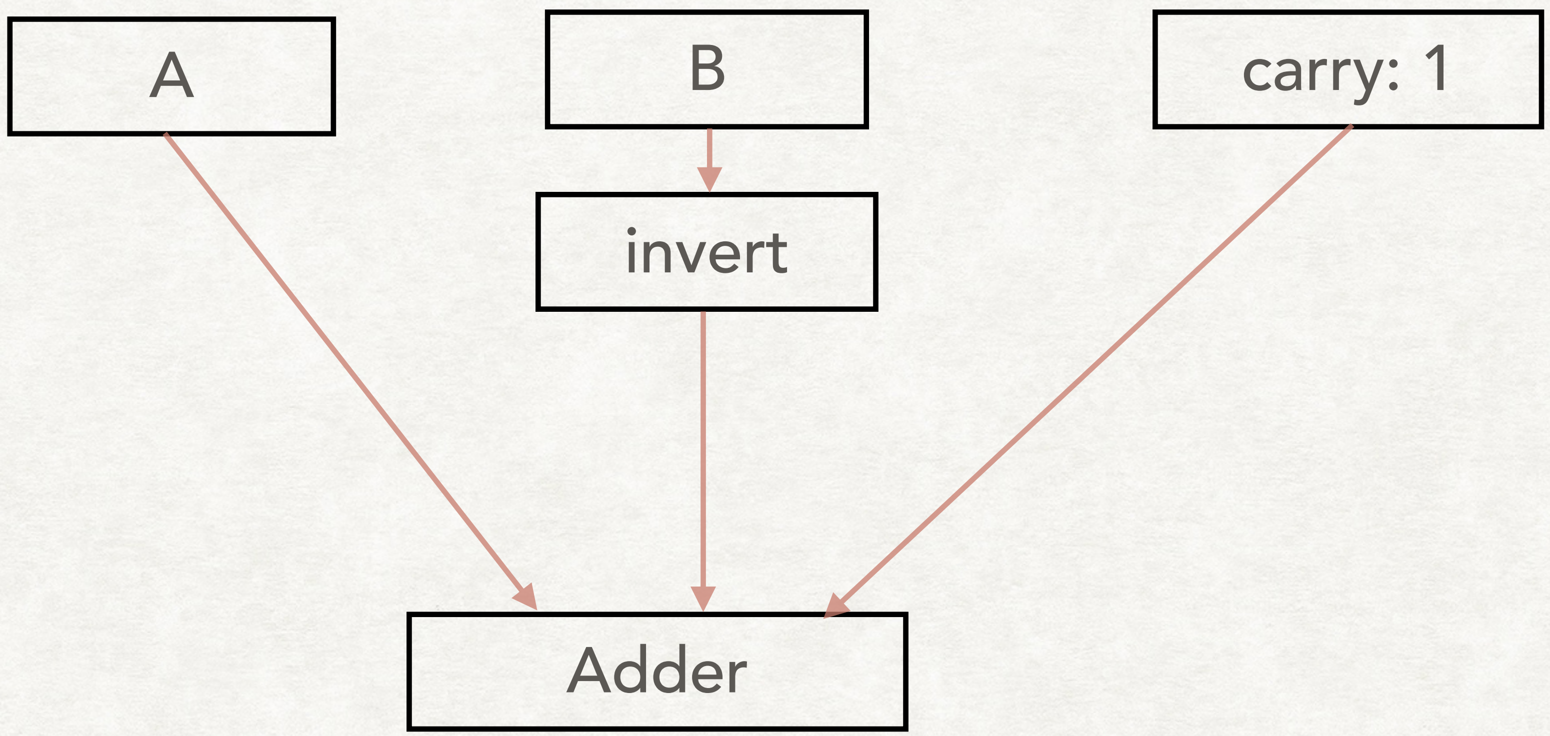


The complete full adder ADD circuit

SUBTRACTION

- We can easily make a subtraction circuit (for two's complement)
  - e.g. **a** - **b**
  - take the one's complement of **b** - what circuit does this?
  - add the answer to **a** (using a carry in to the least significant bit adder of 1)
- Why does this work?

SUBTRACTION CIRCUIT



Output of the adder is A - B

EXAMPLE

- 4 bit numbers
- 6 - 2

6: 0110 2: 0010 C: 1

invert 2: 1101

6	0110
inv 2	1101
carry	1
4	$\boxed{1}$ 0100

The carry out in the 5th column is ignored.

EXAMPLE

- 4 bit numbers
- 5 - (-2)

5: 0101 -2: 1110 C: 1

invert -2: 0001

5	0101
inv -2	0001
carry	1
7	$\boxed{0}$ 0111